CP/M Squares

CP/M Squares
by Rick Kephart January 27, 1988

Not wanting to let the Z80 (CP/M) microprocessor in my Commodore-128 go to waste, I wanted to write some programs for it just so it could be put to use. I did pay for it, after all. This program can be loaded and played using nothing more than the CP/M system disk already supplied with the C-128.

This is a guessing game. The program picks a random square on a checkerboard which you must locate. Enter a row and column as a letter and a number (i.e. the upper left-hand corner square is 'a1' and the lower right-hand corner square is 'h8') and press RETURN, and you will be told if the correct square is above, below, left, or right of that guess. Your guesses will be counted until the correct square is located, at which point you will be prompted to play again, or quit back to CP/M.

Note that though the program is written in Z80 machine language, the program listed here is in C-128 BASIC! This program will produce the machine-language code which the other microprocessor will eventually use, POKEing it into memory. But to use it, it must first be written to a CP/M-formatted disk, which the BASIC SAVE command cannot do. So how do we get the program onto a disk?

This is not as difficult as you might imagine. There is a handy program on side 2 of the CP/M system disk which is just what we need: it saves any section of memory onto a CP/M disk.

The Z80 microprocessor uses BANK1 memory. This is the BANK of memory which will be visible in CP/M mode. Most of this memory remains intact when you BOOT the CP/M disk. So all that needs to be done to transfer our program to a CP/M disk is to POKE it into any safe section of BANK1 memory (here we'll start at 1000 hex or 4096 decimal). BANK1 is used for BASIC variables, but this program uses so few variables they won't even come close to this location, but since it's always good practice to change some the variable-pointers so that the variables cannot overwrite the data, we'll do that here anyway.

As soon as the program has POKEd the ML data into memory, you are prompted to insert side 1 of the CP/M system disk, and the program will BOOT it, to put you into CP/M mode.

Now you are in CP/M mode, with a Z80 game in memory.

Load the SAVE.COM program on side 2 of the system disk with this at the "A>" prompt:

```
A>save
A>save
```

(Note that you type this in twice to start the program). You will be prompted to enter a filename. You may use any name for it, but you must end it with .COM or you won't be able to load and run it.

You will then be prompted for a start address and ending address. The start address, as you might have expected, is 1000. The ending address is 12B3. Here is what it would all look like if you name the program "SQUARES.COM"

```
CP/M 3 SAVE Version 3.0
Enter file (type RETURN to exit): squares.com
Beginning hex address 1000
Ending hex address    12b3
```

Once this has all been done, the program is ready to play! Just enter your filename at the prompt:

```
A>squares
```

And the program will load and run!

I wrote this program without spending any money on CP/M. I used two books I got from a local library to learn to program the chip: "Soul of CP/M" by Michael Waite & Robert Lafore (Howard W. Sams & Co., Inc., 1983), which explains simply and clearly how to program in CP/M, and "A Practical Guide to CP/M" by Carl Townsend (dilithium Press, 1983), which has some very useful charts (including all the opcodes for the mnemonics). Unfortunately, both books only describe 8080 commands, but they were sufficient to write this game.

Not having any assembler, the source code was assembled by hand, and then the machine-language program typed in using the C-128 built-in monitor, and saved using the SAVE.COM program on the CP/M system disk. It was then disassembled using a public-domain Z80 disassembler which I got from a local CP/M BBS.

```
10 POKE 58,16: CLR: BANK 1: PRINT "READING DATA": FOR I = 4096 TO 4787:
   READ A: X=X+A: POKE I,A: NEXT: IF X<>45177 THEN PRINT "ERROR IN DATA
   STATEMENTS": END
20 PRINT "INSERT CP/M SYSTEM DISK": PRINT "PRESS ANY KEY WHEN READY"
30 PRINT "THEN RUN SAVE AND USE 1000 FOR THE BEGINNING ADDRESS":
   PRINT "AND 12B3 FOR THE ENDING
   ADDRESS": GETKEY A$: BOOT
100 DATA 17,47,2,14,9,205,5,0,62,1,50,0,6,197,14,11,205,5,0,183,194,
   31,1,193,121,128,79,4,195,13,1,193
```

```
110 DATA 120,230,7,50,0,4,121,230,7,50,1,4,17,254,4,62,2,18,14,10,205,
   5,0,30,10,14,2,205,5,0,58,0
120 DATA 4,71,58,0,5,214,97,184,202,94,1,17,230,1,210,84,1,17,235,1,
   14,9,205,5,0,62,0,50,254,4,58,1
130 DATA 4,71,58,1,5,214,49,184,202,204,1,245,58,254,4,183,194,123,
   1,17,242,1,14,9,205,5,0,241,17,246,1,210
140 DATA 133,1,17,252,1,14,9,205,5,0,58,0,6,60,50,0,6,254,65,218,151,
   1,201,17,3,2,14,9,205,5,0,6
150 DATA 0,58,0,6,254,10,218,175,1,214,10,4,195,164,1,245,120,198,48,
   95,14,2,205,5,0,241,198,48,95,14,2,205
160 DATA 5,0,30,58,14,2,205,5,0,195,44,1,58,254,4,183,202,138,1,17,13,
   2,14,9,205,5,0,14,1,205,5,0
170 DATA 254,121,202,0,1,201,32,85,112,32,36,32,68,111,119,110,32,36,97,
   110,100,36,32,76,101,102,116,36,32,82,105,103
180 DATA 104,116,36,13,10,71,117,101,115,115,32,35,36,13,10,7,84,104,97,
   116,39,115,32,105,116,33,13,10,10,80,108,97
190 DATA 121,32,97,103,97,105,110,63,40,121,92,110,41,7,36,13,10,10,9,
   32,32,49,32,50,32,51,32,52,32,53,32,54
200 DATA 32,55,32,56,13,10,9,97,124,35,32,35,32,35,32,35,32,35,32,35,32,
   35,32,35,124,13,10,9,98,124,32,35
210 DATA 32,35,32,35,32,35,32,35,32,35,32,35,32,124,13,10,9,99,124,35,32,
   35,32,35,32,35,32,35,32,35,32,35
220 DATA 32,35,124,13,10,9,100,124,32,35,32,35,32,35,32,35,32,35,32,35,
   32,35,32,124,13,10,9,101,124,35,32,35
230 DATA 32,35,32,35,32,35,32,35,32,35,32,35,124,13,10,9,102,124,32,35,
   32,35,32,35,32,35,32,35,32,35,32,35
240 DATA 32,124,13,10,9,103,124,35,32,35,32,35,32,35,32,35,32,35,32,35,
   32,35,124,13,10,9,104,124,32,35,32,35
250 DATA 32,35,32,35,32,35,32,35,32,35,32,124,13,10,9,32,67,80,47,77,32,
   83,81,85,65,82,69,83,32,70,79,82
260 DATA 13,10,32,32,32,32,32,32,32,32,32,32,32,32,84,72,69,32,67,45,49,
   50,56,13,10,10,7,71,117,101,115,115
270 DATA 32,97,32,82,111,119,32,40,108,111,119,101,114,45,99,97,115,101,
   32,108,101,116,116,101,114,41,32,102,111,108,108,111
280 DATA 119,101,100,32,98,121,32,97,32,99,111,108,117,109,110,32,40,110,
   117,109,98,101,114,41,32,97,110,100,32,121,111,117
290 DATA 39,108,108,32,98,101,32,116,111,108,100,32,105,102,32,116,104,
   101,32,67,80,47,77,32,83,81,85,65,82,69,32,105
300 DATA 115,32,85,112,44,32,68,111,119,110,44,32,76,101,102,116,44,32,
   111,114,32,82,105,103,104,116,32,102,114,111,109,32
310 DATA 116,104,101,114,101,46,13,10,10,71,117,101,115,115,32,35,48,
   49,58,36
```

```
                    ORG         0100H

BDOS                EQU         05H
BUFFER              EQU         04FEH
COUNT               EQU         0600H
GUESSCOLUMN         EQU         0501H
```

```
GUESSROW            EQU         0500H
RANDOMCOLUMN        EQU         0401H
RANDOMROW           EQU         0400H

BEGIN:
        LD          DE,INTROMESSAGE

        LD          C,9                 ;print-string: LoaD register C
        CALL        BDOS                ; with 9 and Call (JSR) to BDOS
        LD          A,1
        LD          (COUNT),A           ;keep track of # of guesses
                                        ; starting with A=1
WAIT:
        PUSH        BC                  ;BC will hold 2 Rnd numbers

        LD          C,0BH               ;get console status (H=Hex)
        CALL        BDOS                ; -checks for key-press-
        OR          A
        JP          NZ,CONT             ;Zero-flag set=key-press
                                        ; and break out of WAIT: loop

        POP         BC                  ;random numbers in B and C
        LD          A,C
        ADD         A,B                 ;randomizes number in C
        LD          C,A
        INC         B                   ;randomizes number in B
        JP          WAIT                ; Loop back to WAIT:

CONT:
        POP         BC                  ;Pop random numbers
        LD          A,B                 ;get 1st rnd number into A
        AND         7                   ;must be less than 8
        LD          (RANDOMROW),A       ;store number in memory

        LD          A,C                 ;repeat for second number
        AND         7
        LD          (RANDOMCOLUMN),A

GETGUESS:
        LD          DE,BUFFER           ;prepare buffer for input

        LD          A,2                 ;admit two characters
        LD          (DE),A

        LD          C,0AH               ;read console buffer by putting
        CALL        BDOS                ; 10 ($0A) in register C

        LD          E,0AH               ;print line-feed
        LD          C,2                 ;console output
        CALL        BDOS
```

```
        LD      A,(RANDOMROW)     ;check row guess (letter)
        LD      B,A               ;put correct row in B
        LD      A,(GUESSROW)

        SUB     61H               ;ASCII-letter to number 0-7

        CP      B
        JP      Z,CHECKCOLUMN     ;zero-flag-set=correct row

        LD      DE,UPMESSAGE

        JP      NC,PRINT1         ;Carry-clear=too high
        LD      DE,DOWNMESSAGE

PRINT1:

        LD      C,9               ;print whichever string
        CALL    BDOS              ; was put in DE

        LD      A,0               ;set flag to indicate
        LD      (BUFFER),A        ; wrong row was guessed

CHECKCOLUMN:

        LD      A,(RANDOMCOLUMN)  ;get true column
        LD      B,A               ; and put it in B

        LD      A,(GUESSCOLUMN)
        SUB     '1'               ;ASCII-number to number 0-7

        CP      B
        JP      Z,RIGHTCOLUMN     ;zero-flag-set=correct col.

        PUSH    AF                ;push status word

        LD      A,(BUFFER)        ;correct-row flag
        OR      A
        JP      NZ,NOAND          ;don't print "and" if no
        LD      DE,ANDMESSAGE     ; row direction was printed
        LD      C,9               ;print-string
        CALL    BDOS

NOAND:

        POP     AF                ;get flags back

        LD      DE,LEFTMESSAGE
        JP      NC,PRINT2         ;Carry-clear=too high
```

```
        LD      DE,RIGHTMESSAGE

PRINT2:

        LD      C,9             ;print whichever string is in DE
        CALL    BDOS

COUNTER:

        LD      A,(COUNT)       ;current number of guesses
        INC     A               ;update number of guesses
        LD      (COUNT),A       ;store latest # of guesses

        CP      40H             ;maximum guesses=64
        JP      C,DECIMAL

        RET

DECIMAL:

        LD      DE,GUESSMESSAGE

        LD      C,9             ;print-string
        CALL    BDOS

        LD      B,0             ;B holds number of tens
        LD      A,(COUNT)       ;number to print as decimal

SUBTRACTIONS:

        CP      0AH
        JP      C,PRINTTENS     ;now less than 10

        SUB     0AH
        INC     B               ;count number of tens
        JP      SUBTRACTIONS

PRINTTENS:

        PUSH    AF              ;store units digit

        LD      A,B             ;get tens digit
        ADD     A,'0'           ;convert to ASCII numeral
        LD      E,A             ;print it

        LD      C,2             ;console output
        CALL    BDOS

        POP     AF              ;get units
```

```
        ADD     A,'0'           ;convert to ASCII
        LD      E,A             ;print it
        LD      C,2             ;console output
        CALL    BDOS

        LD      E,':'           ;print a colon
        LD      C,2             ;console output
        CALL    BDOS

        JP      GETGUESS        ;get next guess

RIGHTCOLUMN:

        LD      A,(BUFFER)      ;correct-row flag
        OR      A
        JP      Z,COUNTER       ;zero=wrong row

        LD      DE,CORRECTMESSA
        LD      C,9             ;print-string
        CALL    BDOS

        LD      C,1             ;console input
        CALL    BDOS

        CP      'y'
        JP      Z,BEGIN         ;if input="y" play again

        RET

UPMESSAGE:
        DB      ' Up $'         ; $ means the end-of-string
DOWNMESSAGE:
        DB      ' Down $'

ANDMESSAGE:
        DB      'and$'

LEFTMESSAGE:
        DB      ' Left$'
RIGHTMESSAGE:
        DB      ' Right$'

GUESSMESSAGE:
        DB      0DH,0AH,'Guess #$'

CORRECTMESSA:
        DB      0DH,0AH,07H,'That's it!'
        DB      0DH,0AH,0AH,'Play again?(y',5CH,'n)',07H,'$'

INTROMESSAGE:
```

```
DB          0DH,0AH,0AH,09H,'  1 2 3 4 5 6 7 8',0DH,0AH,09H
DB          'a',7CH,'# # # # # # # #',7CH,0DH,0AH,09H
DB          'b',7CH,' # # # # # # # ',7CH,0DH,0AH,09H
DB          'c',7CH,'# # # # # # # #',7CH,0DH,0AH,09H
DB          'd',7CH,' # # # # # # # ',7CH,0DH,0AH,09H
DB          'e',7CH,'# # # # # # # #',7CH,0DH,0AH,09H
DB          'f',7CH,' # # # # # # # ',7CH,0DH,0AH,09H
DB          'g',7CH,'# # # # # # # #',7CH,0DH,0AH,09H
DB          'h',7CH,' # # # # # # # ',7CH,0DH,0AH,09H
DB          ' CP/M SQUARES FOR',0DH,0AH
DB          '            THE C-128',0DH,0AH,0AH,07H
DB          'Guess a Row (lower-case letter) '
DB          'followed by a column (number) and you'll '
DB          'be told if the CP/M SQUARE is Up, Down, '
DB          'Left, or Right from there.',0DH,0AH,0AH
DB          'Guess #01:$'

END
```

You can write to me at .

rmkq@lphrc.org